

รายงาน

Mini Project Phase 2

วิชา ไมโครโพรเซสเซอร์และระบบคอมพิวเตอร์แบบฝังตัว

( Microprocessor and Embedded Computer System )

เสนอ

อาจารย์ วัชระ ภัคมาตร์

จัดทำโดย

นายโสภณ สุขสมบูรณ์ รหัสนักศึกษา 6201011631188

นักศึกษาชั้นปีที่3 สาขาวิชาวิศวกรรมไฟฟ้า (โทรคมนาคม)

วิชาไมโครโพรเซสเซอร์และระบบคอมพิวเตอร์แบบฝังตัว ประจำภาคการศึกษา 2/2564

สาขาวิชาวิศวกรรมไฟฟ้า(โทรคมนาคม) ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

**Pseudocode**

Global

เริ่มต้น

-ประกาศตัวแปร I , j ,k และ m เป็นตัวแปรชนิด integer

-ประกาศตัวแปร finale\_out เป็นตัวแปรชนิด integer แบบ 8 bit

-ประกาศตัวแปร signed\_ADC , A[16] , B[16] , result เป็นตัวแปรชนิด float

-ระบุขนาดของ CPU สำหรับประมวลผล 16 MHz

adc\_init function

-เปิดใช้งาน ADC mode และทำการ Pre-scale /128

-เลือกใช้ Vref จาก AVCC pin

adc\_convs function

-เปิดใช้งาน ADC Conversion

Timer\_init function

-เปิดใช้งาน Timer0 Overflow Interrupt mode

-กำหนดค่าเริ่มต้นเป็น 0xC2 สำหรับ delay 1 ms

-ทำการ Pre-scale /256

b\_array\_value function

- for loop

- i = 0 , i<15 , i++

-เช็คบิต EEPE ว่าพร้อมหรือไม่

-กำหนด address สำหรับทำ LUT ที่ตำแหน่ง 0x10+i

-เก็บเลข 1 ไว้ใน EEPROM

-เปิดใช้งาน EEPROM write mode

-จบ

-for loop

- j=0 , j<15, j++

-เช็คบิต EEPE ว่าพร้อมหรือไม่

-กำหนด address สำหรับดึงข้อมูลจาก LUT ที่ตำแหน่ง 0x10+i

-เปิดใช้งาน EEPROM read mode

- B[j]= EEDR

Main function

- DDRD เป็น output port

-DDRC เป็น input port

-เซ็ตให้ Port C บิตที่ 5 เป็น 1

-เรียกใช้ฟังก์ชัน adc\_init()

-เรียกใช้ฟังก์ชัน b\_array\_value()

-เรียกใช้ฟังก์ชัน timer\_init()

-เซ็ต interrupt flag (เปิดใช้งาน interrupt)

-while(1)

-จบ

ISR(TIMER\_OVF\_vect)

-กำหนดค่าเริ่มต้น 0xC2

-แปลง ADC จาก unsigned ให้เป็น signed number และเก็บค่าไว้ที่ signed\_ADC

-เรียกใช้ฟังก์ชัน adc\_convs()

-if (PINC5 = 0)

-for loop

-k=15,k>0,k—

-A[k]=A[k-1]

-จบ

-A[0]=signed\_ADC

-result=0

-for loop

-m=0,m<16,m++

-result+=A[m]\*B[m]

-จบ

-แปลงตัวแปร result จาก float ให้เป็น integerแล้วทำค่าไปเก็บที่ finale\_out

-ส่งค่าจากตัวแปร finale\_out ออกไปยังพอร์ต D

-else

-PORTD=0x00

**โปรแกรมภาษาซีและการจำลองการทำงานบน Proteus**

C Code

/\*

\* Mini\_Project\_phase2.c

\*

\* Created: 11/6/2565 18:16:55

\* Author : ASAS

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

*int8\_t* finale\_out=0;

float signed\_ADC;

float A[16],B[16];

float result;

#define *F\_CPU* 16000000 UL

int i,j,k,m;

void adc\_init (void)

{

ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);//set ADC and Prescaler

ADMUX |= (1<<REFS0);//set Vref = AVCC pin

}

void adc\_convs (void)

{

ADCSRA |= (1<<ADSC);//set ADC conversion

}

void timer\_init (void)

{

TIMSK0 = (1<<TOIE0);//set Timer OVF interrupt

TCNT0 = (1<<7)|(1<<6)|(1<<1);//set initial value is 0xC2 for delay 1 ms

TCCR0A = 0x00;

TCCR0B = (1<<CS02);//set prescale is clk/256

}

void b\_array\_value (void)

{

for(i=0;i<15;i++)

{//do Look up table

while(EECR&(1<<EEPE));

EEAR = 0x10+i;//set address

EEDR = 1;//import data in EEPROM

EECR |= (1<<EEMPE);

EECR |= (1<<EEPE);//set write mode

}

for(j=0;j<15;j++)

{//call data in EEPROM

while(EECR&(EEPE));

EEAR=0x10+j;//

EECR |=(1<<EERE); //set read mode

B[j]=EEDR;

}

}

int main(void)

{

DDRD |= 0xFF;

DDRC |= 0x00;

PORTC |= (1<<5);

adc\_init();

b\_array\_value();

timer\_init();

sei();

while (1);

return 0;

}

ISR(TIMER0\_OVF\_vect)//timer OVF interrupt

{

TCNT0 = (1<<7)|(1<<6)|(1<<1);

signed\_ADC = ((ADC/1023.0)\*2.0)-1.0; //change unsigned into signed number

adc\_convs();

if((PINC&(1<<PINC5))==0)//PORTC.5 is high

{

for(k=15;k>0;k--)

{

A[k]=A[k-1];//when new value comes dis old value

}

A[0]=signed\_ADC; //put value from T\_ADC into A[0]

result=0;

for(m=0;m<16;m++)

{

result+=A[m]\*B[m]; //sum of A\*B

}

finale\_out = (int)(result\*127/16); //change float into int

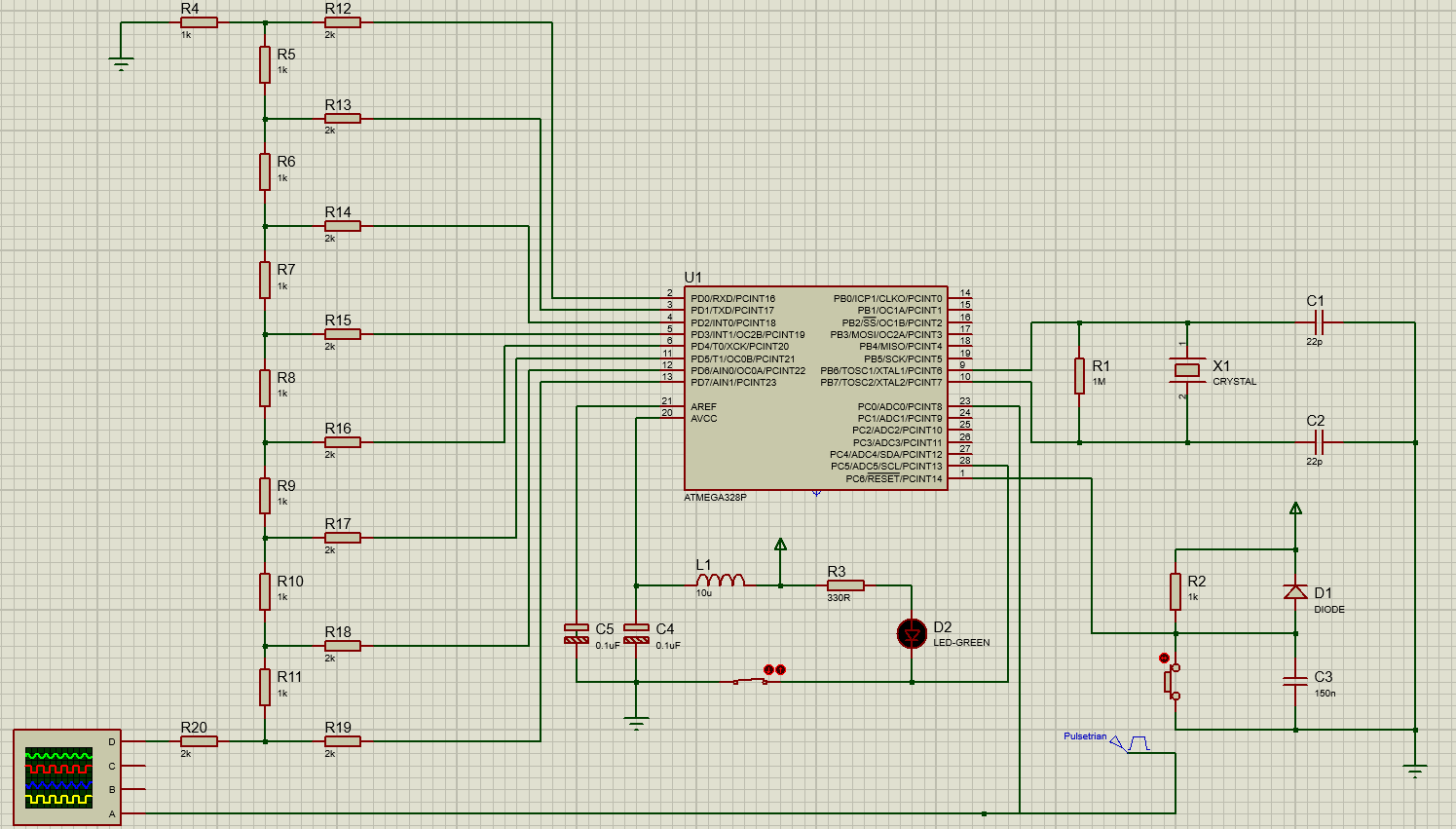
PORTD=finale\_out;

}

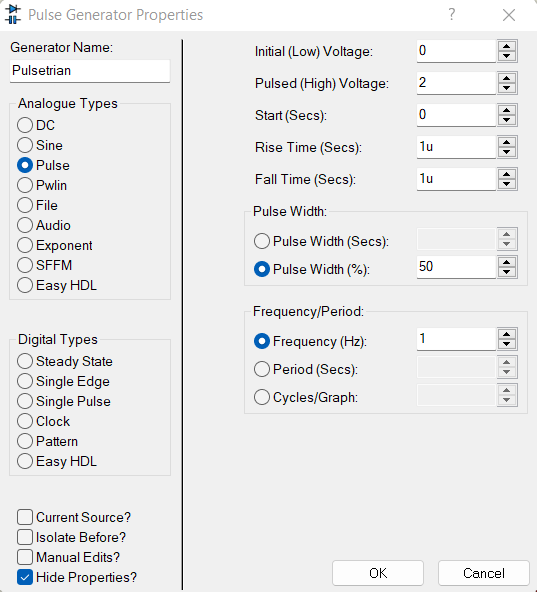
else PORTD=0x00;

}

การจำลองการทำงานบน Proteus

Schematic

(ปรับปรุง Schematic จาก Mini Project Phase 1)



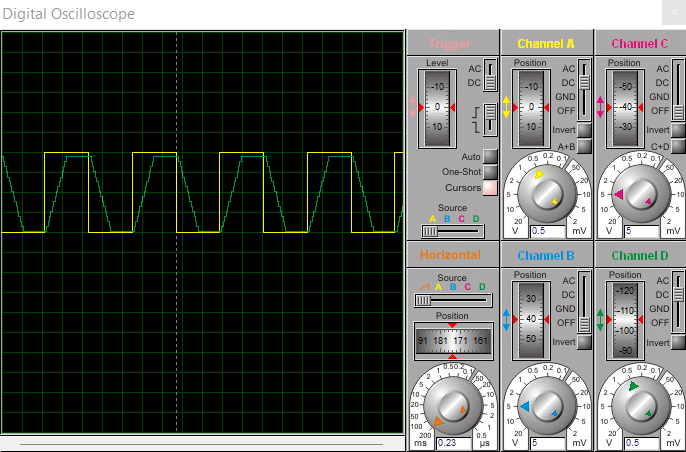
การตั้งค่า สัญญาณ Input

กำหนดสัญญาณอินพุตเป็นสัญญาณ

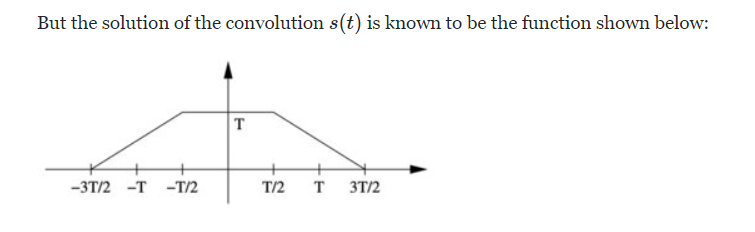
Rectangular Pulse ขนาด 2 V

duty 50% ความถี่ 1 Hz

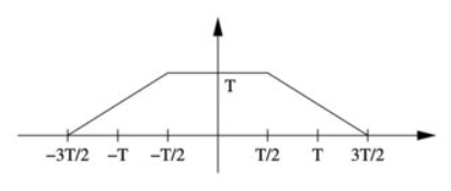
Display



สาเหตุที่สัญญาณ Output ออกมาเป็นสัญญาณดังภาพ (สีเขียว) อันเนื่องมาจากเรานำสัญญาณ Input มาคูณกับ 1 ซึ่งจะได้ค่าเดิม (Input ดั้งเดิม) หลังจากนั้นเรานำผลลัพธ์ที่ได้มาบวกกัน ซึ่งเปรียบเสมือนการทำ Integration โดยผลลัพธ์ที่ได้จะเป็นไปตามทฤษฎีดังนี้



หรือให้คนธรรมดาเข้าใจได้ง่ายคือ สัญญาณ Pulse จะมีช่วง rise และ fall time รวมทั้งช่วงที่มีค่าคงที่ (ความชันเป็น 0) หากเราทำการบวกค่าช่วง rise time ค่าจะเพิ่มขึ้นเรื่อย ๆ อย่างเป็นเชิงเส้น (linearity) ซึ่งเราจะสังเกตจากกราฟของทฤษฎี ในช่วงที่ 1 จะเป็นผลรวมในช่วง rise time ของ rectangular pulse เพิ่มขึ้นอย่างเป็นเชิงเส้น และในช่วง 2 จะเป็นผลรวมที่เกิดในช่วงความชันเป็น 0 และในช่วง 3 จะเกิดจากผลรวมที่เกิดจาก fall time ซึ่งกราฟจะลดลงอย่างเป็นเชิงเส้น เช่นเดียวกันกับในช่วงที่ 1 นั่นเอง ซึ่งเราสรุปได้ว่าผลลัพธ์ที่ได้จากการทำการทดลองนั้นเป็นจริงและถูกต้องตามทฤษฎี



**3**

**2**

**1**